REST APIs

Building your own RESTful API

REST

Representational State Transfer

To understand,

Look at client making a request to server, the server checks if the request can be fulfilled then the request is served or error 404 comes up.

The request is HTTP request

App.get(function(**req**,res))

HTTP is not the only language that they can speak, it can be FTP, SMTP, etc.

HTTPS request is HTTP secured is to make it secure and prevent tapping.

Now if it is a valid request, the server responds with the result, such as data,

App.get(function(req, **res**)) result

The server can fulfill the request by doing computation or they can talk to database or do both.

Building an API is like providing a menu of what requests can be fulfilled.

REST?

REST is kind of architectural style. Others are soap, graphQL, falcor.

Richard N Taylor did REST APIs as his PhD project.

If every web API is made using same principles, then the internet would work more easily and efficiently.

Rules for API:

Use HTTP request verb.

Use specific pattern of Routes/Endpoint URLs.

HTTP Verbs:

GET  
POST  
PUT  
PATCH  
DELETE

Similar to CRUD

GET – READ

POST – CREATE

PUT, PATCH – UPDATE

App.put or App.patch

PUT is basically updating by sending an entire new entry and replacing it with previous one.

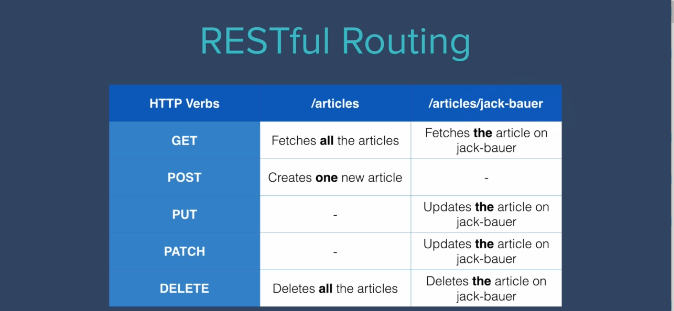
PATCH is basically updating only the piece of data that needs to be updated.

This is same to replacing the entire item we have defected or repairing the only one that is defected.

DELETE- DELETE

Specific pattern of Routes/Endpoint URLs

For example, we can use /elephants routes for serving all elephant resources.



*// This is GET method in which we READ \*all\* articles from our database.*

app.get('/articles', function (*req*, *res*) {

  Article.find({}, function (*err*, *foundArticles*) {

    if (*err*) {

*res*.send(*err*);

    } else {

*res*.send(*foundArticles*) *// Here, we are sending all the data that we have READ from our databse to /articles route (This is similar to how we read from APIs like the weather API)*

    }

  })

})

*// This is tricky because we are not having any kind of front-end elements (like a form element)*

app.post("/articles", function (*req*, *res*) {

  console.log(*req*.body.title); *// we are obtaining title and content from the post request. A post request can be made via Postman API software and key value pairs can be entered in the post request (title, content-key)*

  console.log(*req*.body.content);

  const newArticle = new Article({

    title: req.body.title,

    content: *req*.body.content,

  })

  newArticle.save(function (*err*) {

    if (*err*) {

*res*.send(*err*);

    } else {

*res*.send('success')

    }

  }); *//in this way we can handle the POST requests for our APIs*

})

*// This is the method to handle the DELTE requests for our API*

app.delete("/articles", function (*req*, *res*) {

  Article.deleteMany({}, function (*err*) {

    if (*err*) {

*res*.send(*err*);

    } else {

*res*.send('successfully deleted all articles')

    }

  })

})